Flit: A Bulk Transmission Protocol for RFID-Scale Sensors

Jeremy Gummeson, Pengyu Zhang, Deepak Ganesan
Department of Computer Science
University of Massachusetts, Amherst, MA 01003
{gummeson, pyzhang, dganesan}@cs.umass.edu

ABSTRACT

RFID-scale sensors present a new frontier for distributed sensing. In contrast to existing sensor deployments that rely on battery-powered sensors, RFID-scale sensors rely solely on harvested energy. These devices sense and store data when not in contact with a reader, and use backscatter communication to upload data when a reader is in range. Unlike conventional RFID tags that only transmit identifiers, RFID sensors need to transfer potentially large amounts of data to a reader during each contact event. In this paper, we propose several optimizations to the RFID network stack to support efficient bulk transfer while remaining compatible with existing Gen 2 readers. Our key contribution is the design of a coordinated bulk transfer protocol for RFID-scale sensors that maximizes channel utilization and minimizes energy lost due to idle listening while also minimizing collisions. We present an implementation of the protocol for the Intel WISP, and describe several parameters that are tuned using empirical measurements that characterize the wireless channel. Our results show that the burst protocol improves goodput in comparison to vanilla EPC Gen 2 tags, improves energy-efficiency, allows multiple RFID sensors to share the channel, and also coexists with passive, non-sensor tags.

Categories and Subject Descriptors

C.2.2 [**Network Architecture and Design**]: Network Protocols

General Terms

Design, Experimentation, Measurement, Performance

Keywords

RFID, Bulk Transfer, Sensing

1. INTRODUCTION

A wide range of sensing applications require miniature, ultra-low power sensors in urban and indoor areas. This has led to an increased interest in the design of small, cheap, harvesting-based devices that are attached to common everyday objects (e.g., books, furniture, walls, doors, produce, etc), which can be used for tracking these items [9]. One example of such devices are RFID-scale sensors that exploit ambient light or RF for energy, and use backscatter communication with an RFID reader for data transfer.

RFID-scale sensors, also referred to as Computational RFIDs or CRFIDs, present a new frontier for distributed sensing [4]. These devices are distinct from existing battery-powered sensor platforms as well as commercial RFIDs. They are designed for continuous sensing, however, unlike existing continuous sensing devices (e.g. Motes), they use small capacitor buffers, rely solely on energy harvesting, and use more power-efficient backscatter communication for data transfer. CRFIDs are also distinct from commercial RFIDs in that they use hybrid harvesting to enable continuous sensing, computation, and storage rather than just vanilla identification.

In this paper, we investigate how to efficiently utilize the energy buffer of an energy harvesting CRFID node for burst message exchange. We focus our attention on mobile CRFIDs [21], whose movements result in two communication states: connected and disconnected. While tags traverse their environment they perform a series of sensing and computation operations. As time elapses, these devices buffer some amount of data during disconnected operation; occasionally they encounter a reader, resulting in a variable connection interval during which a tag may offload buffered data.

Our goal is to optimize the bulk transfer of this buffered data from the CRFID sensor to an RFID reader. Because CRFID sensors have small energy buffers, it is imperative that communications maximize goodput while minimizing the amount of energy per unit data. This presents several challenges. First, commercial RFID readers follow the EPC Gen 2 protocol which is optimized for large numbers of tags that each transfer a small amount of data (tag identifier). This protocol is inefficient when considering sparsely deployed CRFID sensors that each potentially need to transfer large amounts of buffered data to a reader. EPC Gen 2 also makes duty-cycling for CRFIDs very difficult to implement because they must listen to a potentially large number of messages while waiting to transmit; this is unacceptable as CRFIDs treat energy as a precious commodity. While a complete re-design of the protocol stack is possible, this would mean that CRFIDs could not take advantage of existing commercial RFID readers, making them far less attractive for widespread use. Rather, we seek to support efficient bulk data transfer while still being compatible with commercially available EPC Gen 2 RFID readers. Second, when compared with other sensing platforms, CRFIDs have different hardware components, use different energy sources, use different energy buffers, and follow a different communication protocol. Thus, designing an energy-optimized bulk transfer protocol for RFID sensors requires an entirely new set of bandwidth and energy-optimization mechanisms. Third, CRFIDs present different usage scenarios since they are largely deployed indoors, and often on mobile objects or people. Thus, any data transfer protocol should operate effectively under scenarios where there are short contact durations with readers, and considerable changes in link characteristics during mobility.

Our protocol, Flit, provides a fast and efficient alternative to the existing EPC Gen 2 protocol for bulk data transfer from sensors, while still remaining compatible with existing RFID readers. Flit makes three fundamental changes to the protocol stack. First, it enables each sensor to transfer data in a burst by responding to all slots in a query round rather than just its assigned slot. This design choice improves goodput and energy-efficiency by reducing wasted slots, and takes advantage of extended query rounds with less control overhead. Second, Flit coordinates across sensors by using explicit burst notifiers that are echoed by RFID readers, rather than devices randomly picking a slot in which to transmit. This approach serializes burst transfers across nodes, thereby allowing greater goodput while reducing potential for collisions. Third, Flit improves energy-efficiency by duty-cycling the RFID sensor when another CRFID is in the middle of a burst. This avoids wasted energy due to overhearing of reader messages during the burst, thereby enabling better use of a small buffer of stored energy on the CRFID sensor.

Our results show that:

- Flit achieves 60% greater goodput than EPC Gen 2 for a single tag at different distances from the reader, and for different mobility conditions. A breakdown shows that much of these gains are due to the use of larger query rounds, and avoiding wasted slots in the round.
- Flit achieves 4.5x more goodput than an EPC Gen 2 tag when three tags are transferring data concurrently, and 9.2x goodput when five tags are transferring simultaneously. In addition, Flit has considerably higher fairness than EPC Gen 2, which is skewed towards the node with highest SNR to the reader. This allows sensors to take advantage of shorter contact durations with readers.
- Duty-cycling in Flit achieves up 6.04x better average power efficiency than a non-duty cycled implementation. These power savings are acheived by minimizing energy consumpted from listening to other CRFIDs' transmissions.

2. AN EPC CLASS 1 GEN 2 PRIMER

The Gen 2 protocol for RFID tags is designed to inventory large tag populations over a number of communication rounds. To realize this protocol, an RFID must traverse a simple state machine and respond appropriately to a set of reader commands. Throughout this discussion, refer to Figure 1 to understand how a sequence of reader commands and tag responses are used to transmit data to a reader. The critical subset of EPC commands a CRFID must implement are:

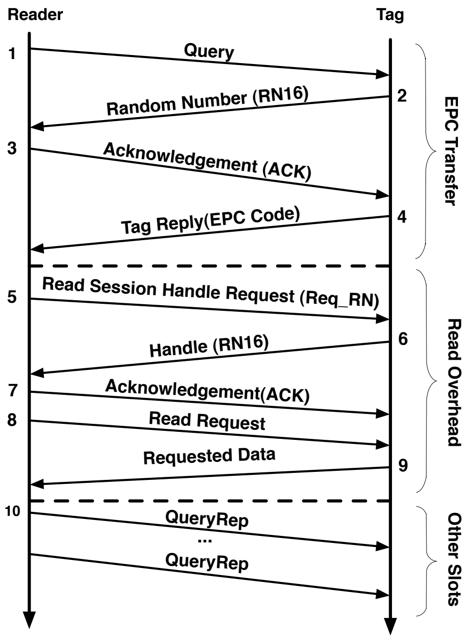

Figure 1: A series of message exchanges are required between a reader and tag to read the tag's EPC code or user memory.

Query, QueryRep, and QueryAdjust: A *Query* message (1) initiates a round of communication. This message specifies several round parameters. The most critical of these parameters is Q, which defines the number of slots in a round to be $2^Q - 1$ where $0 \leq Q \leq 15$. Tags generate a non-negative, random slot counter within the range specified by Q. The reader chooses Q such that collisions between tags are minimized.

Slots after a *Query* are occupied by *QueryRep* (10) and *QueryAdjust* messages. *QueryRep* messages indicate a successive slot for this round; *QueryAdjust* messages indicate a successive slot and additionally adjust the current Q value by +/- 1. After receiving either message, the tag decrements its slot counter; when the counter reaches 0, the tag proceeds.

Ack(RN16): To disambiguate tags in the event of collision, an *RN16* (2) message is used. The RN16 is a 16-bit value randomly generated by the tag. Upon decoding an RN16 from tag(s), the reader will echo one of the RN16s it received as an ACK

EPC: A tag knows it was chosen for communication if the received ACK matches the sent RN16; if this is not the case, the tag gives up on this round of communication to avoid further collision. After receiving its own RN16, the tag may backscatters its *EPC* (4) code to the reader. After sending its EPC, the tag will not respond to subsequent QueryReps or QueryAdjusts during this round of communication.

Req_RN: A reader that wants to further investigate a tag's state may send a *Req_RN* message (5). This message establishes a 16-bit handle to be use for subsequent communication. The tag echoes the handle back to the reader as an ACK (7).

Read: After establishing a session handle, the reader may send a *read* command (8) to request a segment of the tag's memory; after receiving a Read command, the tag responds with the requested data (9). The tag appends the session handle and a two byte CRC computed across the payload.

3. LIMITATIONS OF GEN 2 FOR CRFIDS

In this section, we discuss several limitations the EPC Gen 2 protocol has on designing an energy-efficient bulk data transfer protocol that transfers data from a CRFID sensor to a reader. To understand these limits, we conducted a benchmark study that quantifies the timing and energy requirements of relevant Gen 2 messages for the Intel WISP 4.1; the results of this study are presented in Table 1. For each message type, we report the time required to finish sending or receiving a particular message in the *Active* column. The amount of time between a message and subsequent message is reported in the *Idle* column. We compute the energy for a particular operation by multiplying the platform power consumption by the sum of the *Idle* and *active* durations. For each message type, we also note whether WISP sends(TX) or receives(RX) the message, as the WISP consumes more power when receiving a message because it increases its clock frequency.

3.1 Singulation Inefficiency

The Gen 2 RFID protocol is designed around inventorying large numbers of tags that need only report a static identifier. It is therefore primarily focused on collision avoidance for a large number of passive tags. In this section, we show that EPC Gen 2 is inefficient for bulk transfer both in terms of throughput and energy-efficiency.

A key parameter that controls the efficiency of the EPC Gen 2 protocol is the window size, Q. The window size is a parameter that is set by a reader based on the tag population that it observes, as described in §2; during a round a number of slots are chosen such that the probability of collision between two tag responses is negligible. The EPC Gen 2 standard provides some general guidelines as opposed to a specific algorithm for how to set Q, so the implementation of the algorithm is vendor specific and is typically unavailable to the customer. In addition, there is often no way to control the Q values set by a reader since modern RFID readers are designed for ease of use and hide low-level protocol parameters from the operator. In particular, the Impinj Speedway reader we use in this paper offers no visbility into the chosen Q value; the resulting window size is completely decided by the reader's proprietary algorithm.

The efficiency of the EPC protocol depends on the value of Q set by the reader in each round. For example, if a reader picks $Q = 3$ and there is only one tag present, then there are 8 slots in this round, including the Query, one of which is utilized by the tag. In addition to the obvious throughput inefficiency, this is also inefficient energy-wise as a tag incurs the energy overhead of listening to the QueryRep or QueryAdj messages for the slots that it does not respond to.

To understand the practical inefficiencies of Gen 2 singulation, we looked at the round lengths selected by an Impinj Speedway reader when a single WISP tag is placed in front of it. While the reader does not provide an interface to obtain the chosen Q value, the WISP is programmable, therefore we were able to obtain the numbers by transmitting this information in place of the EPC code. Figure 2 shows that the reader typically chose a Q value between 1 and 6, with a mean of 2.5; this behavior held for distance up to 7 m. These Q values indicate that the number of slots in a round varies between 2 and 64 slots despite only a single tag being present, clearly a major source of inefficiency.

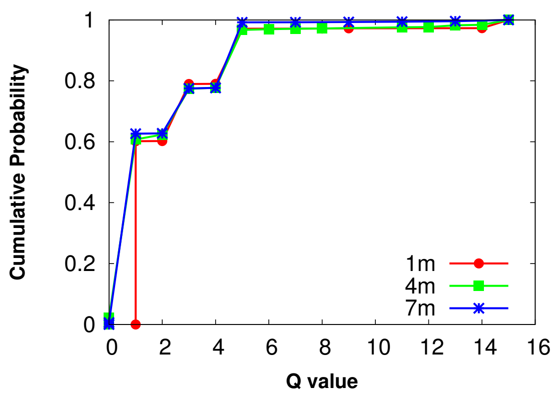

Figure 2: This plot shows what Q value a reader actually chooses when non-burst communication is used

Operation	# bits	Time Active	Time Idle	Energy
Query(RX)	22	983 μs	52 μs	648 nJ
QueryRep(RX)	4	273 μs	50 μs	210 nJ
QueryAdj(RX)	9	415 μs	51 μs	319 nJ
Read(RX)	52	2100 μs	50 μs	1615 nJ
RN16(TX)	16	641 μs	2390 μs	422 nJ
Ack(RX)	18	660 μs	36 μs	508 nJ
Req_RN(RX)	40	1616 μs	51 μs	1241 nJ
EPC(TX)	128	2450 μs	2360 μs	1615 nJ
CRC16	–	452 μs	–	307 nJ

Table 1: A CRFID emulates Gen 2 in software leading to widely varying amounts of energy consumption depending on the command.

To further drive this point, we refer to Table 1. Based on a mean Q value that varies between 2 and 6 as in Figure 2, the extra communication slots result in degradation throughput that varies between 9.7 - 294.2% and energy consumption that increases by between 34.9 - 546.4% as compared to a single tag communication during a single slot round. It is also important to note that this is a lower bound on the amount of energy required as CRFIDs will likely remain in an active state between received messages. These performance penalties change as a function of Q, which is in turn a function of the number of tags present.

3.2 Inefficiency of Read Messages

Gen 2 supports tag user memory operations in addition to simple EPC queries. Of particular interest is the Read command, which allows a reader to request a region of the tag's user memory. While at first glance, Read message seem to ideal for transmitting sensor data from a tag, we show that they are inefficient in terms of channel utilization and energy consumption.

Read messages are attractive because they support variable response lengths; a long read message could potentially overcome the singulation inefficiencies we previously highlighted, in addition to allowing CRFIDs to transmit large amounts of data to a reader. In theory, large Read messages are possible since Gen 2 specifies that an upper limit of 255 bytes on their size, but in practice, the size of read messages is limited by factors such as bit error rate, hardware limitations, and timing drift. For example, we found that the Impinj Speedway reader supports Read requests of lengths upto 60 bytes. For the Intel WISP, we found that read error rates sharply approached 100%, when 16 bytes of data were requested via Reads. Our hypothesis is that these practical limitations stem from three reasons: a) long messages are vulnerable to high bit-error rate (BER) at longer distances, particularly since the path loss on a backscatter link drops as the fourth power of distance [20], b) longer messages incur more timing drift, and RFID-scale devices often do not have real-time clocks to adjust for these, and c) large messages incur high memory overhead, which is limiting for RFID-scale devices. On the Intel WISP, both BER at higher distances and the timing drift were issues that made it difficult to get longer Read messages across to the reader.

Read messages also incur significant control overhead, which results in considerable throughput and energy inefficiency. As seen in Figure 1, a tag needs to be singulated prior to handling a read request, and depending on the Q value chosen for the round, may need to listen to several slots before it can set up a Read with the reader. This design clearly outlines the priorities of EPC Gen 2 — it is designed for obtaining identifiers from tags, and Reads are a second-class citizen that is intended to be used sparingly. The overhead is compounded by the fact that long Read messages are not practical, and is inefficient energy-wise since the tag is forced to listen to a long series of messages before its turn.

3.3 Lack of Duty-Cycling Support

Another major limitation of EPC Gen 2 is its lack of support for duty-cycling. While duty-cycling of a CRFID may seem unimportant for communication since the device receives power from the reader, this is not entirely true. The distance at which a CRFID can communicate with a reader is far more than the distance at which a tag can receive power from a reader. It is for this reason that passive tags have operating distances of a few feet from a reader, whereas a hybrid-powered CRFID (RF + ambient harvesting) or a battery-powered active tag can have communication ranges of 50-70 feet [10]. At longer distances, a CRFID needs to duty-cycle and leverage low-power states since they are using precious reserves of stored energy or are operating on small amounts of ambient power.

Wireless MAC protocols designed with duty-cycling in mind typically use a number of mechanisms to synchronize senders and receivers, and buffer packets while waiting for synchronization to occur. For example, the 802.15.4 MAC layers uses preambles to synchronize sleeping senders and receivers, the 802.11 power save mode (PSM) relies on the access point buffering, and TDMA MACs have fixed slots, allowing a device to sleep for a fixed duration without the risk of missing messages. In contrast, EPC Gen 2 has non-deterministic arrival times of messages and variable round lengths. While Q determines the length of a round, this length is often not set at the beginning of a round. Instead, Q can be dynamically changed using QueryAdjust messages (based on estimated tag density), and a tag would not know the current value of Q if it misses a QueryAdjust message. In addition, slot lengths can be different since slots can terminate at different times due to timeouts after different steps of the protocol. The consequence of lack of duty-cycling support is that Gen 2 can cause CRFIDs to waste excessive energy on idle listening while waiting for their communication slot.

4. FLIT DESIGN

There are a number of factors to consider when designing a bulk data transfer mechanism for Gen 2. Such a mechanism must strive to: 1) maximize data transfer rates so that sensor tags can transfer their data quickly and efficiently to a reader during short contact events, 2) minimize power consumption so that a CRFID can maximize the amount of data transferred using its small energy buffer, and 3) interoperate with standard commercial RFID readers, so that CRFIDs can leverage existing RFID reader infrastructure.

To realize these goals, we present the design of a burst protocol for CRFID sensors. First, we discuss the design tradeoffs in using an EPC Query versus the Read command as the data transfer primitive. Next, we demonstrate how sensors can achieve high levels of goodput using burst-mode data transfer that leverages unused slots in the EPC Gen 2 protocol. Third, we show a coordination mechanism that uses burst notifiers to avoid collisions among bursting tags. Fourth, we present a duty-cycling mechanism that minimizes the energy lost to idle listening. Finally, we discuss implications of the design choices that we make when there are a mix of sensor tags that are bursting and standard EPC Gen 2 tags that are only transmitting their identifier.

4.1 Read vs EPC for Burst Transfer

The first question in designing a burst data transfer protocol is which EPC Gen 2 message primitive to use as the building block for transferring data. Two options present themselves in terms of adapting the EPC Gen 2 protocol for bulk data transfer from the sensor to the reader. The first option is to use EPC Read command which allows a variable amount of data to be transmitted from a tag to reader, but has several inefficiencies as described above. The second is to use the *EPC* message, and send application data instead of the 12 byte static identifier within this message.

We first look at the energy efficiency of Reads vs EPC messages using the set of energy benchmarks in Table 1. The energy efficiency of the read command varies with the length of the data sent in response to the read request, while an EPC message is always 12 bytes. These benchmarks were captured using the Intel WISP 4.1 [16] (more details in §5).

From this breakdown, we compute the amount of energy consumed per byte of data transfer. Each Read command incurs energy overhead for steps 1–8 in Figure 1 that precede the read payload. The energy consumed for each EPC command varies a small degree based on whether it is in response to a Query, QueryRep or QueryAdjust since they have different sizes.

Suppose that EPC codes are used to transfer data to the RFID reader. An analysis based on a round with 4 slots (Q

= 2) will result in 12 bytes of data arriving at the reader per slot, for a total of 48 bytes of data. This process takes 39.77 μs.

Now, suppose that a Read message is used to request data from a tag's user memory. Since we must now use the EPC data to singulate an individual tag for the subsequent Read command, a larger Read message is required to compensate for this additional overhead. When considering the same 4 slots as in the EPC based approach, the Read command incurs the previously computed time delay as overhead, as well as steps 5 - 8 from Figure 1. In order to match the throughput of a pure EPC-based approach under this scenario, a Read request of at least 116 bytes is required. This result indicates that Reads are a poor choice for high throughput tag to reader communications based on the reasons outlined in §3.2. A similar analysis of the energy required per byte of transmitted data shows that a Read with 143 byte request size is required to match the energy efficiency of the pure EPC approach. A similar conclusion applies if energy efficient tag to reader communication is needed.

One advantage of Gen 2 Reads, is that they have built in options for security. After a tag receives the handle message depicted in Figure 1, the tag may optionally be sent an *access* message that contains a password; reception of a correct password moves the tag into a logical state called *Secured*. This state may be utilized to protect portions of tag memory targeted by a subsequent Read command. However, since computational RFIDs can implement cryptography, they could instead encrypt data locally if a particular application requires it.

4.2 Burst-Mode EPC Transfer

Having selected the 12 byte EPC message as the building block for bulk transfer, we turn to the question of improving efficiency when several hundreds of bytes of data need to be transferred using this message primitive. If the Gen 2 protocol were followed, data transfer would need to be over several tens or hundreds of rounds, and a CRFID would receive only one slot in each round. As described in §3.1, this would be extremely inefficient due to poor choices of Q at the reader.

The central idea in burst transfer is to ignore Gen 2 semantics of rounds, and to treat the protocol simply as a sequence of unassigned request/response slots. Each of these slots can be initiated by a Query, QueryRep, or QueryAdjust, but the burst protocol does not treat them differently. Instead, a CRFID sensor assumes that every slot is available to it for burst transfer, and just transfers its data in a sequence of consecutive slots. Before discussing issues of coordination across multiple tags (§4.4), we look at the benefits that this offers to a single tag.

The key benefit of burst transfer from a single tag is that we are no longer limited by poor selection of Q by a reader (§3.1). In fact, we turn a drawback into an advantage. To obtain the full benefits of burst transfer, we want the reader to choose a large Q. As previously shown, a round is initiated by a Query message for the first slot, and the other $2^Q - 1$ slots are initiated by QueryReps. A few slots are initiated by QueryAdjust messages, whose purpose is to increment or decrement Q in the middle of a round. A subtle benefit of QueryReps and QueryAdjs, as opposed to Queries, is their brevity. Based on protocol specs, Query, QueryRep and QueryAdjust messages have lengths of 22, 4, and 9 bits respectively. As Q grows, the energy expended during a round of communication becomes dominated by round trips involving reps and adjusts.

The energy and throughput benefits of using longer Q are quantified in the following equations:

$$\mathrm{E_{round}} = \mathrm{E_{query}} + \left(2^{Q+A} - 1\right) \cdot \mathrm{E_{rep}} + n \cdot \ \mathrm{E_{adjust}} \quad (1)$$

$$\mathrm{Goodput_{rnd}} = \frac{12}{\mathrm{T_{query}}} + \frac{12 \cdot \left(2^{Q+A} - 1\right)}{\mathrm{T_{rep}}} + \frac{12 \cdot n}{\mathrm{T_{adj}}} \quad (2)$$

Equation 1 shows the total energy spent on a round of communication, which includes the energy spent on listening and replying to the first query slot($\mathrm{E_{query}}$), the energy spent listening/replying to subsequent slots initiated by QueryReps (Q is the initial value assigned by the reader, and A indicates how it was adjusted during the round), and finally the energy spent on n QueryAdj messages that were sent during the round. Equation 2 shows the goodput counterpart, which takes into account the length of an EPC message (12 bytes), and the time for the three types of queries.

Using numbers from our microbenchmarks in Table 1, we see that a long round with $Q = 15$ can give about 10% benefit in both energy and goodput over a short round with $Q = 0$.

In summary, treating the Gen 2 protocol as a sequence of unassigned slots enables us to a) limit inefficiency due to empty slots caused by poor selection of Q, and b) improve efficiency by taking advantage of shorter slots initiated by QueryRep messages.

4.3 Coordination via Burst Notifier

Responding in every slot has a severe limitation: if multiple CRFIDs are present, they will suffer from collisions and see reduced energy efficiency and goodput instead of the improvements. An active radio system could solve this problem using control messages such as RTS/CTS or an overhearing-based approach such as CSMA to coordinate transfers between peer nodes. These approaches are not suitable for backscatter communication circuits because they are unable to decode messages transmitted by peers. An alternative would be for the reader to explicitly select a tag in the Query or QueryRep message, and all other tags that receive the message can ignore the slot. However, as mentioned earlier, QueryReps are only *4 bits* long, and leaves no room for such addressing. Besides there is the limitation that readers do not allow modifications of Query messages, making any such approach impractical. Thus, we ask the question: *How can CRFID sensors use the existing EPC protocol to efficiently coordinate bursts?*

A closer look at the Gen 2 Query/EPC exchange reveals that there is a two-way handshake being performed, which presents a solution to this problem. As shown in Step 3 of Figure 1, the reader echoes the RN16 of the RFID it chooses to occupy a given communication slot. Our strategy is to overload the RN16 to signify that a particular CRFID is currently bursting. We accomplish this by providing a special interpretation of a segment of reserved RN16s; we partition the space of RN16s as $0 < n < 2^{16}$, where n is the number of CRFID sensors deployed and values less than n are considered burst notifiers. The value of n is statically selected at compile time and is chosen based on the maximum number of CRFIDs envisioned for a particular application. A sensor that wishes to send a burst of EPCs will use its statically selected burst notifier chosen from the available pool, instead of a random value. Note that the sensor selects a notifier just once for an entire burst, rather than once per slot as is done by a standard tag.

The RN16 burst notifier is used in the following way: prior to initiating a burst, a CRFID sensor listens to the channel after decoding a query, rep, or adjust message. If the sensor observes an Ack within the range of burst RN16s, it should remain silent to avoid colliding with an ongoing burst. If the slot contains an RN16 outside of this range, it can go ahead and start a burst transfer after the current slot using its own burst notifier, as non-burst EPC messages occupy only one slot.

It is, of course, possible that another CRFID sensor is in the middle of its burst and either the reader might have missed the burst notifier or the listening sensor may not have received the notifier echoed by the reader due to channel error. Both cases would lead the listening sensor to conclude that the channel is free and start to burst, resulting in collisions at the reader. A collision at the reader typically results in the reader receiving the stronger signal among the colliding tags due to capture effect. The reader echoes the burst notifier that it receives, which results in only the sensor with stronger signal continuing to burst. While a collision could also result in neither signal being received by the reader, we handle this case by assigning a random back-off interval after hearing no Ack when one is expected.

To prevent the sensor from holding the channel indefinitely, the burst will terminate after a small, fixed amount of time that is large enough to amortize coordination overheads, but small enough to allow mobile tags with limited communication opportunities a chance to offload a burst of data to the reader.

4.4 Duty-cycled Coordination

Burst transfer is a natural fit for CRFID duty-cycling for two reasons: a) transfer is in large chunks of consecutive slots, enabling other nodes to sleep for longer durations and re-charge while waiting for a burst to end, and b) inefficiencies incurred due to duty-cycling such as wasted slots because a tag is asleep or wasted energy for listening because it is awake too early can be amortized over the longer sleep durations.

While bursts are convenient for duty-cycling, the lack of enough bits in the Query/QueryRep messages impacts duty-cycling efficiency as well. If a waiting tag knew precisely how much longer a burst from another tag would last, it could sleep for exactly that duration. However, this information is unavailable since a sensor tag relies on the burst notifier from the reader to detect a burst, which provides no information on the time remaining for the burst. Thus, a tag needs to periodically wakeup to check the channel and detect if a burst has ended. Thus, a key challenge for a duty-cycling strategy is to efficiently find the end of a burst so that sensors can capture the channel from another sensor between bursts and react quickly to mobility dynamics while avoiding most of the energy wastage caused by overhearing.

Thus, there are two questions that remain regarding how to duty-cycle an CRFID sensor: a) the amount of time a sensor should probe the channel and b) how much time a sensor should sleep. Since tag-to-tag communication is impossible,

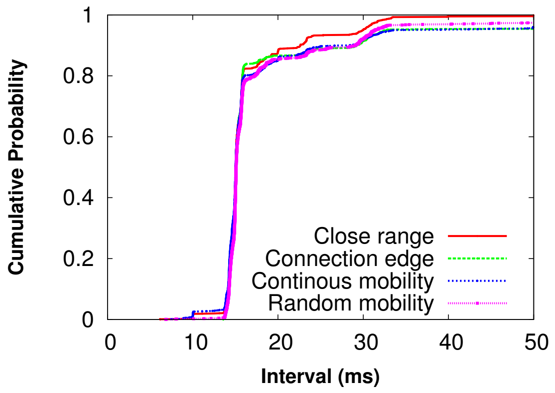

Figure 3: Most Query, QueryRep, and QueryAdj messages have inter-arrival times of less than 20 ms.

we do not consider adaptive policies for determining these intervals since the new probe and sleep intervals would need to be shared with all tags. We now describe how these intervals should be statically selected.

Probe Duration: The probe duration should be long enough such that a tag can detect whether another tag is continuing to burst. This duration is equivalent to a single slot in a query round. A sensor tag wakes up, listens to the first Query, QueryRep, or QueryAdjust slot, and sees whether a burst notifier is echoed by the reader during this slot. If so, it concludes that another tag is bursting and goes to sleep; if not, it concludes that it can initiate its own burst and starts transmission in the next slot. In the middle of a burst, if a tag detects that the reader has echoed a different burst notifier it concludes that another CRFID sensor is bursting and goes to sleep to save energy.

A potential issue here is that the duration of a slot can vary because a) Query, QueryRep, and QueryAdjust messages are of different lengths, b) a slot can terminate at different times depending on whether the reader times out after the RN16, Ack or EPC steps in its state machine and c) mobility can introduce additional dynamics. To address this, we look at the probe duration empirically by measuring the inter-arrival time of Query, QueryRep or QueryAdjust messages for a continuous exchange between an Intel WISP programmed with the EPC Gen 2 protocol and a reader. We look at this distribution for different distances from the reader, as well as for different mobility patterns. Figure 3 shows the CDF of the inter-message duration. The results show that the inter-query intervals do not depend significantly on the distance, and are impacted a little ,but not a lot, by mobility. The knee of the curves is in the 15-20 ms range, thus we select 20 ms as our probe duration; this probe duration is short and provides a reasonable guarantee that a query will be heard by the WISP during the period it is awake.

Sleep Interval: There are several considerations in determining the sleep interval. First, the sleep interval must be long enough that we get significant energy benefits from duty-cycling. Second, it should be short enough that a tag can quickly react to mobility-induced channel dynamics. Third, it should have sufficient randomization so that

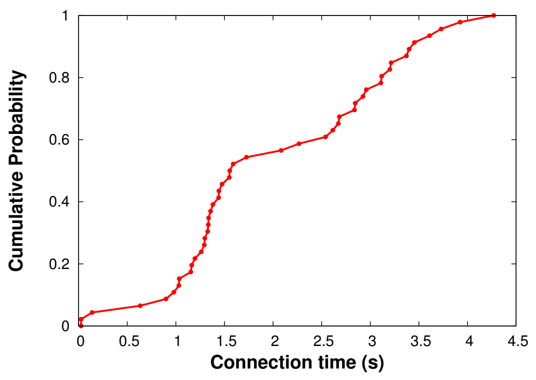

Figure 4: For human-scale mobility rates, the connection time between a tag and reader typically lasts several seconds.

we avoid unwanted synchronization issues that can result from multiple tags waking up at the same time.

In terms of the energy consumption, we want a duty-cycle of lower than 10%, hence the sleep duration should be at least 200 ms when the probe duration is 20 ms. To understand the typical contact duration at walking speed, we use 1 reader in a corridor, and walk in circles around it. We found that a typical contact duration is a few seconds in duration (see Figure 4), hence the sleep duration should be much smaller than this number. To prevent synchronization issues, the tag can randomize the sleep time within a tolerable range that provides a desired amount of energy savings while maintaining reactivity to expected mobility patterns for a given deployment.

4.5 Coexistence with Non-burst Tags

While our discussion thus far has assumed the tag population comprises solely of sensor tags that have to transfer data in a burst, we now look at the implications when a mix of sensor tags and standard EPC Gen 2 tags are communicating with the same reader infrastructure. Not surprisingly, the net effect is that standard EPC Gen 2 tags incur more delay in communicating with a reader infrastructure. However, there are mitigating factors that can enable better coordination across tags.

The burst mode transfer mechanism that fills up all slots of a round impacts standard Gen 2 tags in two ways. First, a standard tag which picks a slot within a round will collide with a burst tag, resulting in loss of one of the messages. In practice, we find that because of reader sensitivity, the reader gets one of the CRFID's messages with high probability (due to capture effect), hence it is still possible that the standard tag gets its data through. However, if the burst tag has the stronger signal, the standard tag suffers. Second, the burst transfer approach results in large Q values, which makes a round long; since standard tags only respond in one slot within a round, this makes their response slow. This effect is mitigated by the fact that we limit bursts to a relatively short duration of time (1sec in our implementation), after which a sensor goes to sleep for a short window of time before bursting again. The duration between bursts is sufficient for a few short communication rounds, enabling standard tags to get their data through.

The use of burst notifier facilitates co-ordination across

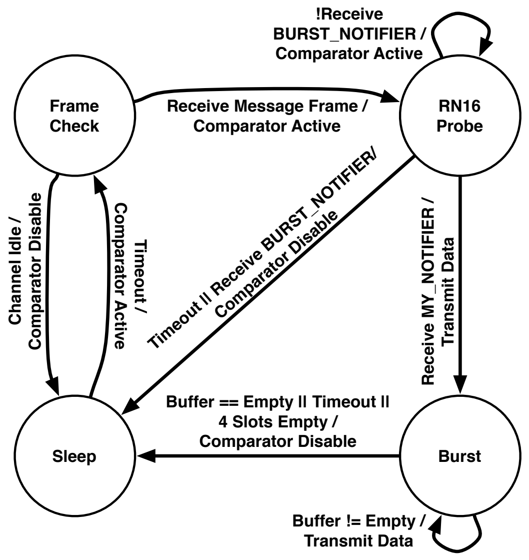

Figure 5: A coordinated bursting protocol for CRFID sensors can be implemented as a state machine. The protocol uses sleep states to both avoid contention and achieve energy efficiency.

CRFIDs, however, passive tags are free to choose any value from 0 to 2^{16} as its RN16, so it is possible that a passive tag could choose an RN16 that conflicts with a burst notifier. However, we choose a small part of the space for burst notifiers since we expect the number of sensors in the vicinity of a reader to be in the tens (equivalent to the number of objects in the vicinity of a reader) as opposed to thousands. Thus, the probability of collision is low. In addition, the RN16s are chosen anew in each round, hence a standard tag would likely choose a non-colliding RN16 in the next round.

5. IMPLEMENTATION

Our bulk transmission protocol is well suited for implementation on CRFID sensors because it is a modification of the EPC Gen 2 protocol they already support. CRFID sensors that want to implement the protocol need only modify their state machine to properly handle burst-mode transmission, burst notifers, and duty cycle appropriately in response to received message frames from a reader. In this section, we show the state machine we used to implement our Bulk Transmission Protocol for the Intel WISP. Next, we describe how state machine parameters can be defined based on results from channel measurements and mobility experiments. Finally, we give some details that describe the evaluation methodology that drove the design of Flit.

Figure 5 shows the state machine used to implement our Bulk Transmission Protocol for the Intel WISP. Sensors that have data to send initialize a timer interrupt and begin operation in the *Sleep* state. After this timer expires, the sensor activates its comparator and enters state *Frame Check*; after initializing another timeout value, the microcontroller enters a low-power mode, only waking up to handle an incoming message frame. Upon receiving a valid message frame, the sensor will enter state *RN16 Probe*; else, if the sensor does not hear a valid delimiter from a reader, it goes back to state *Sleep* after timing out. While in state *RN16 Probe*, the sensor initializes its timer with another timeout value; after hearing at least one empty frame from the reader, in which the sensor does not hear another sensor's BURST_NOTIFIER, it will send its own BURST_NOTIFIER in response to a Query, QueryRep, or QueryAdjust message. If the sensor hears its own BURST_NOTIFIER, it enters state *Burst*; if another sensor's BURST_NOTIFIER is heard, instead of an empty slot or if the timeout value is reached, the sensor enters state *Sleep*. Upon entering state *Burst*, the sensor will again initialize a timer, then begin transferring the contents of its buffered data as an EPC message in response to Query,QueryRep, or QueryAdjust messages; the sensor uses its BURST_NOTIFIER to send every message within the burst. Upon completion, timeout, or detecting 4 slots during which it finds no acknowledgement, the sensor returns to state *Sleep*.

5.1 Parameter Selection

While the state machine we previously described is a useful framework to constuct our protocol, implementation of the state machine was not straightforward, and needed several parameters to be carefully chosen and implementation aspects to be carefully addressed. We describe a few of these challenges in this section.

Timeouts: The timeout values used in our state machine are chosen based on Figures 3 and 4 in §4 that give good insight into expected connection intervals and message interarrival times respectively. In practice, these timeout values are used as comparison values for Timer A_1 on the WISP's MSP430 microcontroller. When considering hardware constraints and initialization overheads, one must also be careful to not choose a set of timeout values that generate too many interrupts that interfere with the WISP's ability to timely respond to reader messages. In practice, timeouts > 2 ms give the WISP sufficient time to listen for messages, while also providing the time needed to for timer initialization.

Duty-cycling: To implement the state machine, we also need to understand how the RF subsystem operates, and how to achieve maximal duty-cycling benefits. The RF subsystem comprises two components: a) the analog comparator that senses the channel to detect the presence of a bit, and b) the microcontroller that wakes up upon each interrupt from the comparator to process the bit and check if a valid message is present. The duty-cycling strategy is straightforward — shutting off the comparator avoids any energy lost from responding to interrupts and idle listening.

Burst Length: We choose one second as the length of a burst since it is long enough to obtain substantial duty-cycling benefits. After using up a burst, a tag pause 250 ms before trying to capture the channel for another burst. This duration provides a window for other tags to capture the channel or passive tags to transmit their identifier.

Burst Notifiers: The final parameter we consider is the burst notifier used by tags to coordinate their burst transfers. When modifying the WISP firmware, we found it can be difficult to get the state machine to stay within the tight timing constraints specified by EPC Gen 2 protocol. Complex operations in the firmware diminish responsiveness and in the end manifest as a reduction in goodput. For example, choosing a poor ordering of comparisons while looking for a burst notifier can lead to a 30% reduction in goodput. We also found that messages sent from the reader to the WISP can contain bit errors; one example is the RN16 field, which in actuality contains only 15 bits of consistent data. Thus, a careful implementation was needed to make burst notifiers operate correctly.

5.2 Debugging and Evaluation Methodology

Implementing and evaluating our system on the Intel WISP was particularly challenging due to the limited visibility, extremely low-power nature, and tight timing constraints. Any logging on the device would dramatically change performance and the nature of bugs, therefore all our evaluation had to be performed external to the device. We highlight how we overcame these challenges below:

Visibility: To overcome visibility issues, a WISP can be tethered to a JTAG debugging tool to observe its internal state. In many mobile scenarios, JTAG tethering is not a viable option; in these cases, software state of interest can be transmitted to a reader as an EPC code. To understand timing-related phenomena, Timer A1, which is unused by the firmware, can be used to capture the intervals which again are sent as EPC messages. In many cases, we found initializing a timer took long enough to violate the timing requirements of EPC Gen 2 and resulted in the reader being unable to decode tag message frames. To solve this problem, we use a Telos mote [2] to count the time between two GPIO interrupts generated by the WISP and log the resulting timing information TinyOS-2.x [13] application. The Telos platform is well-suited for this scale of timings; the platform itself does not significantly impact mobility because of its small size.

Energy Limitations: Evaluating a burst mode transfer mechanism with a limited energy supply is difficult. Because our work assumes tags have some amount of buffered energy when initiating contact with a reader, it would be difficult to use a capacitor-based buffer while allowing for repeatability. To remove measurement dependencies on harvested power, we power tags directly with batteries; this setup assumes they have a plentiful store of energy while in contact with a reader. Separately, we collected energy benchmarks that measured the energy required for sending and receiving protocol messages and other computation overhead. Combining these two approaches in our evaluation, we were able to evaluate the duty-cycling aspects of our protocol.

Channel Measurements: Another evaluation challenge lies in being unable to measure the channel directly. The protocol used to communicate with the Impinj reader, LLRP, does not provide an interface to directly obtain information about the PHY layer. This makes it impossible to get a completely accurate picture of channel characteristics and bit-error rates. One way to capture low-level channel measurements would be to use a passive EPC Gen 2 packet sniffer that listens to message exchanges between a tag and reader. A USRP-based Gen 2 channel monitor [7] has been implemented for this very purpose, but we found that the channel conditions observed at the monitor were much different than those observed by the tag.

To overcome these obstacles, we embed counters into EPC messages that indicate the number and type of messages received. These message counters assist in determining the messages sent in the forward link, as well as their contribution towards data delivery. Because an EPC code transfer happens directly after receiving ACK messages from the reader, the ratio of ACKs to EPC codes received at the reader gives a measure of the losses in the backscatter link. Additionally, the counter information we send allows us to align messages with GPIO timing triggers on the Telos mote.

6. EVALUATION

In this section, we evaluate the implementation of our bulk transmission protocol. The evaluation consists of four parts: 1) quantifying the goodput achievable by burst-mode EPC transfer, 2) showing that our burst notifier based coordination mechanism retains most of the goodput achieved by bursts by avoiding collisions, 3) demonstrating the energy benefits of duty-cycling, and 4) evaluating the interaction between bursting tags and standard EPC tags.

All evaluation results are obtained empirically; we feel this is important because of the complex RF environment that affects tag performance. We present results for small numbers of CRFIDs due to the limited number of prototypes available; the class of applications we consider (described in §1) are well aligned to the number of tags we use. In all of our experiments we use an Impinj Speedway reader with a fixed set of configuration parameters. The reader paramters used were: 1) Type A reference interval (Tari) = 25.0 μs, 2) Pulse Interval encoding (PIE) = 2.0:1, 3) Forward link = PR-ASK, 4) Pulse width = 0.5 5) Link Frequency = 256 KHz 6) Reverse Modulation = Miller 4, 6) Transmit power = 30.00 dBm, 7) channel = frequency hopping.

6.1 Burst mode transmission

The burst mode transmission protocol that we described in Section 4 ensures that all slots created for a round of communication are utilized by CRFIDs. In this section, we evaluate our burst transmission protocol in three ways: 1) We measure the window size allocated by Impinj reader when burst mode transmission is utilized, 2) We evaluate the goodput benefit gained by burst mode transmission, and 3) We provide a breakdown to show where the goodput benefits comes from.

Window size: In §4.2, we argued that the burst mode transmission strategy results in an RFID reader choosing a large window size (Q value) within a round of communication, leading to greater opportunities for using shorter messages, such as QueryRep or QueryAdjust, for data delivery. To validate this argument, we design an experiment that compares the Q value chosen by an Impinj reader for standard EPC transfer vs burst transfer. Our experiment setup places an Intel WISP in line-of-sight of an Impinj reader's antenna. The WISP piggybacks the Q value in place of the EPC code that it backscatters to the reader. We found that the Impinj reader consistently selects Q $\approx$ 10 for burst transmission independently of distance. As described in §4.2, a large Q value is good for burst transmission. In contrast, the Impinj reader selects $Q \approx 2$ when the standard EPC Gen 2 protocol is used. Any Q value larger than 0 leads to un-utilized slots, therefore, this choice results in a significant fraction of wasted slots.

Goodput: To quantify how burst-mode transfer of EPC codes better utilizes slots, we design an experiment that compares the goodput of a burst-optimized version where a single tag responds within every slot versus EPC Gen 2. In this experiment, we measure the goodput of an Intel WISP tag programmed to act as a conventional EPC Gen 2 tag vs a WISP programmed for burst mode operation. We log the average throughput observed by the reader at different distances for several minutes and compare the results. Figure 6 shows that burst mode communication achieves 60% higher goodput than standard EPC Gen 2, and that these benefits are sustained across different distances. It is also notable that, in practice, the benefits of burst transfer are considerably larger than those that we predicted in §4.2. Our hypothesis is that additional gains are a result of the reader over-allocating slots for passive tags, resulting in comparatively low goodput.

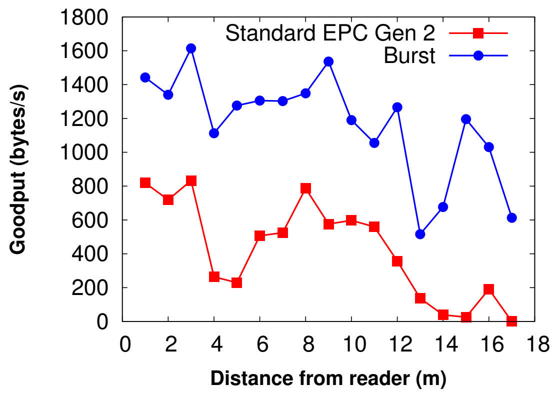

Figure 6: A CRFID sensor can achieve a 60% improvement in goodput by utilizing all slots in a communication round.

Goodput breakdown: The increase in goodput for burst transmissions stems from two factors. First, we utilize every slot rather than one slot in each inventory round to transmit data. As shown in Figure 7, a standard EPC Gen 2 protocol only utilize only 64.8% of slots in each inventory round at 1m and 68.0% at 7m since it responds only to either a Query message or a QueryRep message, but not both – the other slots will go unutilized. In contrast, bursts will utilizes 100% of the slots. Second, in burst mode, we get more opportunities to exploit the shorter inventory messages: QueryReps and QueryAdjusts. Shorter messages in the forward link have lower loss rates; this leads to tags successfully capturing slots for data transfer with higher probability. For large Q, communication is dominated by slots that are initiated by QueryRep and QueryAdjust messages; as shown in Figure 7, the percentage of QueryReps at 1 m while bursting increases to 76.6%, as compared to the 34.2% slots in standard EPC Gen 2. This result holds true at a distance of 7 m as well. By exploiting QueryRep and QueryAdjust messages, tags get more opportunities for data transfer that contribute to higher goodput.

6.2 Coordinating bursts

While burst-mode transfer provides significant improvements to goodput, it also introduces problems when multiple RFID sensors wish to use the channel simultaneously. In this experiment, we evaluate how much the co-ordination mechanism benefits goodput when multiple tags are transmitting.

We consider a baseline case when a single CRFID is present, a low-contention case when three CRFIDs are simultaneously transferring data and a high-contention case when five

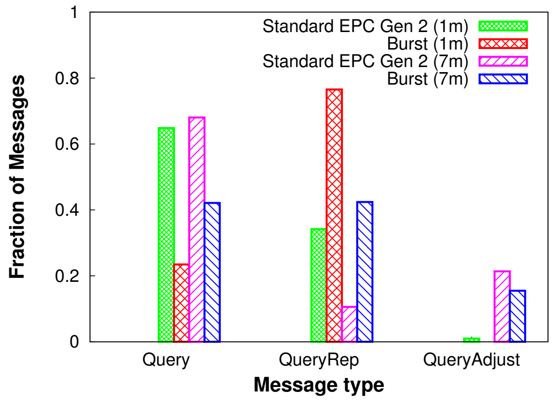

Figure 7: Bursting CRFIDs cause the number of round slots to increase. We observe this as an increase in the fraction of QueryReps received.

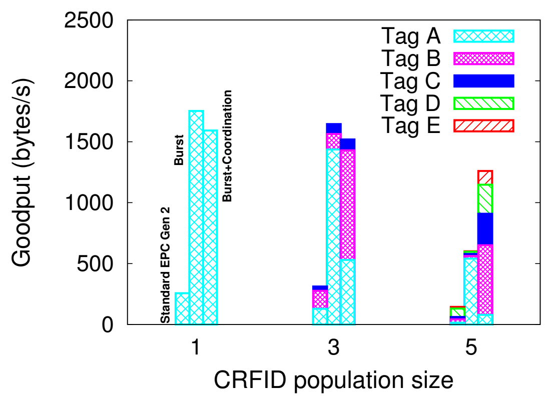

Figure 8: Coordination improves throughput and fairness as more bursting CRFIDs compete for the channel. For small tag populations, coordination incurs a small protocol overhead.

tags are simultaneously transferring data. The number of tags in our experiment is limited by the WISPs that we have available, however, we expect that the number of sensors transferring simultaneously will be a relatively small number. All tags are placed in a line with their antennas placed parallel to the Impinj reader's antenna within line-of-sight at a distance of 1 meter. Figure 8 shows a breakdown of the goodput for three protocols: a) standard EPC Gen 2, b) Burst mode without coordination, and c) Burst mode transfer with cooordination.

First, we look at the case where there is a single tag. We see that the standard EPC Gen 2 tag performs significantly worse than the bursting sensor tags, as expected. We also see that the co-ordination scheme performs about 9.2% worse than the case without co-ordination. This is because of the overhead required for coordination. After finishing a burst transmission, a coordinated tag releases the channel for a 50 ms to allow other waiting tags to acquire the channel, which reduces goodput. Next, we increase the tag population to three; we see a similar behavior in terms of overall goodput, but the split across nodes is very different. Overall, we see that burst mode transfer with coordination is still a little lower (7.7%) in total goodput than the uncoordinated case. However, the un-coordinated case gives out more than 87.3% of the channel to one of the three tags. The burst protocol with co-ordination is considerably fairer — all three tags receive a good chunk of the overall goodput. Finally, when the tag population reaches five, the impact of collisions becomes significant and adversely impacts the performance of uncoordinated sensors. For bursts without coordination, the total goodput reduces by 65.7% as compared to the single tag and the three tag cases. In addition, the uncoordinated scheme is highly unfair and allocates 90.0% of the goodput to one of the five tags. In contrast, when the five tags coordinate, they acheive similar goodput as observed in the one and three node cases.

Table 2 shows the Received Signal Strength Indicator (RSSI) values logged at the reader for the backscattered data from the five tags and their respective goodputs. Despite having similar distance from the reader, the tags observe different RSSI values as a result of differences in antenna orientation. As a result the tag(Tag A) with strongest RSSI(-39 dBm) captures the channel entirely and achieves a goodput of 540.5 bytes/second. The other four tags cumulatively get 10.0% of the total goodput, each achieving less than 30 bytes/second. In contrast, coordination results in a more even partitioning of goodput. Table 2 shows the coordination mechanism doesn't necessarily favor the tag with highest RSSI — in fact although Tag D's RSSI is lower than Tag A, it gets a large fraction of the goodput.

ID	Coordination		No Coordination	
	RSSI (dBm)	Goodput	RSSI (dBm)	Goodput
A	-41.7	78.9 Bps	-39.6	540.5 Bps
B	-37.7	571.1 Bps	-45.1	21.5 Bps
C	-39.9	258.8 Bps	-47.2	16.9 Bps
D	-46.0	238.2 Bps	-45.9	17.8 Bps
E	-50.1	112.8 Bps	-54.9	4.0 Bps

Table 2: A breakdown of the goodput from Fig 8 for the 5 tag case, shows that the amount of goodput a tag achieves is related to the average RSSI value at the reader.

6.3 Coordination-Aware Duty-Cycling

Coordination allows tags to avoid collisions between each others bursts, but does nothing to prevent energy consumed due to idle listening. We now evaluate our duty cycling mechanism which duty-cycles the comparator to reduce energy consumed due to idle listening. Our evaluation answers three questions: a) how much power is saved due to duty-cycling?, B) does duty-cycling result in degradation in goodput? and C) how does duty cycling affect the distribution of burst length among tags?

To quantify duty-cycling benefits, five tags continuously transmit EPCs for 10 minutes while deployed in a linear topology several feet from the reader with their antennas perpendicular to the reader's antenna. We set the probe and slew durations as defined in §4.4 as 20 and 200 ms respectively and look at the power consumed by each tag.

Figure 9 shows that duty-cycling reduces the average power consumption by 3.24x to 6.04x across the five tags when considering the amount of time the MCU spends in sleep vs

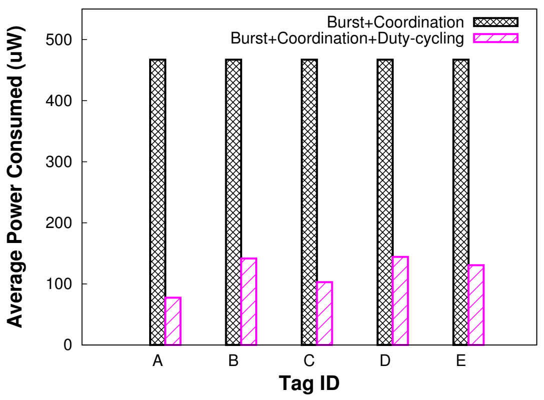

Figure 9: The Average power consumption of a CRFID is reduced considerably by reducing the energy lost to idle listening.

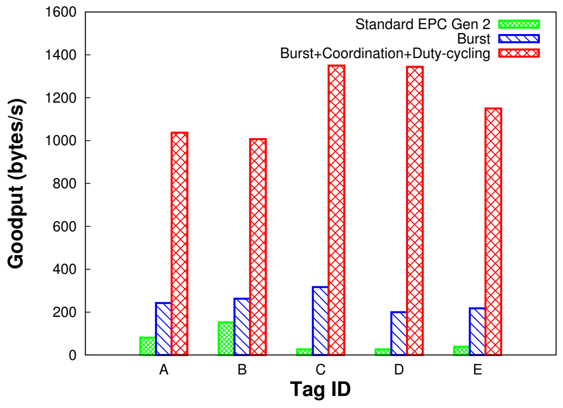

Figure 10: Nodes that duty cycle achieve considerably higher goodput. This is a result of reduced channel contention while non-bursting tags sleep.

active modes. The benefits of duty cycling differ depending on how often an individual tag gets access to the channel.

While duty-cycling improves overall energy consumption, a natural question is whether these gains come at the cost of goodput. Figure 10 shows that duty cycle based coordinated bursting tags have the highest goodput. This demonstrates that our duty-cycling mechanism does not sacrifice goodput to achieve its energy gains because of reduced channel contention that makes burst notifiers even more effective.

Finally, to better understand the dynamics of coordination and how frequently nodes switch among each other, we look at a breakdown of the duration that each tag holds the channel for the same dataset. We define a burst period as a contiguous segment when a single tag is bursting; at the end of the period, some other tag acquires the channel and starts bursting. These results are plotted in Figure 11. The results show that bursts are variable in length, even though they are allowed a maximum length of 1 second.

Fragmentation of bursts can result in unfair channel sharing because tags with better placement will see fewer spurious losses and hold the channel longer more frequently. We see this phenomena manifest prominently in Figure 11, as Tag A was placed in an unintentionally poor orientation. Although Tag A's bursts are shorter on average, they maintain throughput (Figure 10 – when the channel becomes poor according to tag A, another node grabs the channel. Upon hearing another tag's burst notifier, tag A immediately goes to sleep, resulting in very high power efficiency.

6.4 Coexistence with passive tags

In this section, we investigate the interaction between our burst protocol and standard EPC Gen 2. We answer two questions when both bursting tags and standard EPC tags are transmitting to an Impinj reader: 1) How does the goodput of CRFIDs change when they compete for the channel with multiple standard EPC tags? 2) What is the time between inventorying standard EPC tag when CRFIDs are bursting? We setup experiments where both bursting tags and passive tags are collectively within the field of an Impinj Reader's antenna. We measure the goodput achieved by the tags that run the Flit protocol, and the interval required to inventory passive Gen 2 tags.

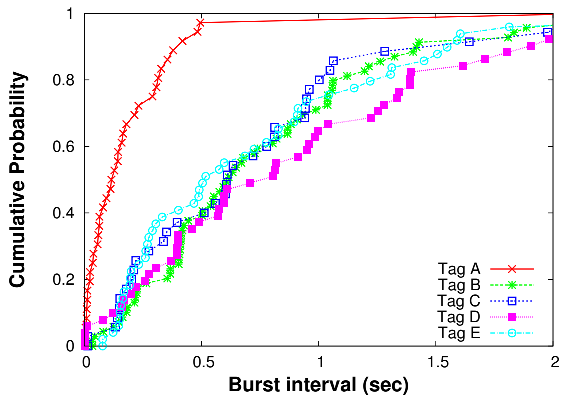

Figure 11: When tags use coordination, they achieve similar burst lengths; in this case Tag A is an exception because of particularly poor antenna orientation.

Goodput: Our goal in this experiment is to understand how increasing passive tag populations impact the goodput of bursting CRFIDs. Since we did not have enough Intel WISPs to use as passive tags, we use commercial passive tags for this experiment. We deploy five CRFIDs that are continually bursting to a reader, and increase the commercial passive tag population from 5 to 30. All commercial passive tags and CRFIDs are placed 1 meter from reader and spread in a line parallel with the reader antenna. Table 3 shows the average goodput as the passive tag population increases. The results show that there is only a small effect until about 20 tags (the average goodput is 253.3 bytes/second which is only a bit lower than 264.4 bytes/second when there are no passive commercial tags). For larger populations of passive tags, there are more collisions in slots which impacts goodput of burst CRFIDs. However, we see that despite a relatively large passive tag population, CRFIDs continue to perform well while bursting.

Time between inventory: In this experiment, we look at how the time to inventory a population of passive tags is impacted by the presence of a burst CRFID. We consider two variants of Flit — one with the standard 50 ms sleep

# Passive Tags	Avg Goodput (B/s)
0	264.4
5	222.1
10	238.0
15	270.7
20	253.3
25	184.2
30	184.9

Table 3: Deploying passive tags alongside a bursting CRFID causes the goodput of the CRFID to degrade as the number of passive tags increases.

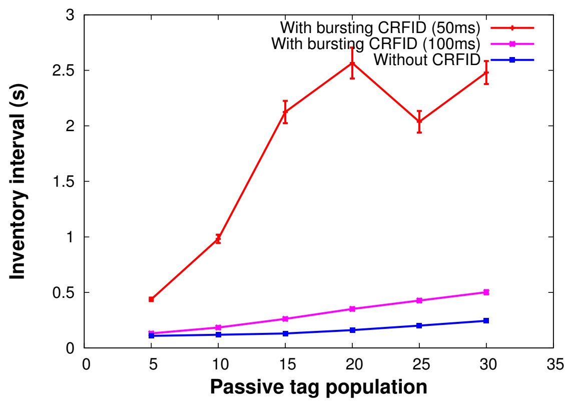

Figure 12: The average interval for a passive tag to be read increases with the tag population.

period between bursts, and one where this duration is increased to 100 ms. Figure 12 shows the average inventory time and associated 95% confidence intervals when there's a population solely comprising commercial passive tags vs a mix of passive tags and bursting CRFIDs. While the average inventory time increases with increasing population, the increase is greater when a bursting CRFID is in the mix. However, the total time is still of the order of a few seconds, showing that passive tags still see opportunities to get data through. In addition, increasing the sleep duration between bursts to 100 ms dramatically reduces the inventory time to be only slightly larger than the case when there are only passive tags. This provides a simple knob in deployments where inventorying time for passive tags needs to be low.

7. RELATED WORK

Empirical Wireless Measurements: In recent years there has been significant work in measuring the characteristics of wireless communication channels for a variety of communication mechanisms (e.g. [18]). Perhaps most relevant to our work is [6], which quantifies wireless performance of Gen 2 through an empirical study that looks only at messages sent by a reader. In contrast, our focus is on improving tag to reader communications for CRFIDs. Additionly, our measurements provide visibility into the backscatter link by piggybacking statistics in EPC codes.

Bulk Data Transfer: The bulk transmission of data through a wireless channel has been studied in a variety of contexts including hardware and software systems. In an 802.11 setting [14] describes several mechanisms that together provide a transport layer optimized for bulk data transmission. Our work looks at RFID backscatter as opposed to 802.11; specifically, we focus on optimizations at the MAC layer for improving the throughput of bulk data transfer. Also of interest is [12], which provides a bulk transport protocol for 802.15.4 based wireless sensor networks. This work uses an end-to-end acknowledgement-based protocol to provide reliability, a rate control mechanism to minimize transfer time and a metric derived from combined signal strengths to avoiding hidden terminal issues. Instead, we focus on detecting other CRFID bursts using reader messages rather than explicitly avoiding them with a collision avoidance metric. Most similar to our work is [8], which proposes the use of persistent read handles in EPC Gen 2 to offload large amounts of data from an individual tag. While more efficient than requiring singulation for each read command, this approach is limited to data transfer from a single tag and cannot take advantage of the relatively shorter QueryRep and QueryAdjust commands.

While Flit provides bulk data transfer at the MAC layer, it does not preclude performance improvements at the PHY layer as defined by EPC Gen 2. In [22], the authors present a system that optimizes the channel and bit-rate selected by a reader, such that tag goodput is maximized. This approach is completely compatible with Flit; a CRFID communicating with an optimized reader would only see further goodput gains.

The elimination of idle slots introduced by coordination is not a new idea and has been looked at in wired contexts, where a common data bus is arbitrated for use by multiple entities. One good example of this is the bus parking mechanism used in the PowerPC 60x [3]. Here, the bus arbiter speculatively grants a bus master before receiving a master request; this grant message is coordinated by broadcasting to all attached devices. This is a very similar concept to the coordination mechanism we use in Flit, where the reader broadcasts a burst notifier to all tags within range.

Energy Management: There has been some recent work on energy management for CRFID systems. The work presented in [15] instruments code at compile time to enable checkpointing software state to non-volatile storage; the goal is to avoid energy wasted on work that was lost to power outages. In [5], a run-time system is presented that adaptively schedules a task to avoid energy wastage. Wastage is defined as work that does not complete due to of energy limitations, as well as harvested energy that cannot be stored. [10] looks at tradeoffs when ambient harvesting is used with RF harvesting, and explores the choice of hardware components to satisfy application requirements given an anticipated amount of harvested energy. Our work is complementary to these efforts, as we improve the bandwidth and energy-efficiency of the communication stack and efficiencies would only further improve with a run-time management strategy.

Looking beyond CRFID research, there has been substantial work on energy management in harvesting-based sensors. For example, [17] achieves perpetual operation by scheduling tasks to match predicted energy harvesting rates, [11] and [19] looks at adaptive duty-cycling strategies for harvesting-based systems, [9] looks at using efficient solar harvesting in combination with an ultra-wideband impulse radio to balance energy usage at an even smaller scale, and commercial efforts have looked at micro-energy harvesting from miniature solar panels, thermal differences, vibrations, and wireless power-over-distance technology (e.g. Powercast [1]). Such techniques may be useful to ensure a suitable amount of buffered energy for bursts is available during reader contact periods, and is complementary to this paper.

8. CONCLUSION

In this paper we presented the design, implementation, and evaluation of Flit, a bulk transmission protocol for RFID-scale sensors. Through a careful analysis of the EPC Gen 2 protocol for passive RFIDs, we identified several opportunities for improvements to both goodput and energy efficiency when considering small numbers of CRFIDs that have large amounts of data to send. Through empirical evaluation, we demonstrated that significant gains in goodput are possible over a variety of distances when compared to a tag that implements vanilla EPC Gen 2. To enable the simultaneous bulk transfer of data from multiple CRFIDs, we designed a simple coordination mechanism that works well in practice and through an experimental evaluation, showed the complete system retains most of the performance improvements we observed for a single, uncoordinated CRFID. Finally, we demonstrated that our protocol can coexist with passive RFIDs can, but are inventoried with increased latency.

While flit is an unreliable MAC protocol, Flit provides an excellent building block for a reliable transfer protocol. A simple window-based reliable protocol over Flit would work as follows: After each burst of messages from the sensor to the reader, the reader sends a read command that has a bitmap of the messages that were received by the reader. In the next burst, the sensor can re-transmit these missing frame in addition to adding other data. We are currently implementing this protocol over Flit.

9. ACKNOWLEDGMENTS

This work was funded by NSF grants CNS-0916577, CNS-0855128, and CNS-0923313 . We thank Intel Research Seattle for providing several WISPs; Our shepherd Guoliang Xing for providing guidance in preparing our final draft; and the anonymous reviewers for their insightful comments.

10. REFERENCES

[1] Powercast website. `http://www.powercastco.com`.

[2] TelosB Datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`.

[3] M. Allen, M. Alexander, C. Wright, and J. Chang. Designing the powerpc 60x bus. *Micro, IEEE*, 14(5):42, oct. 1994.

[4] M. Buettner, B. Greenstein, A. Sample, J. R. Smith, and D. Wetherall. Revisiting Smart Dust with RFID Sensor Networks. In *HotNets*, October 2008.

[5] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: An energy-aware runtime for computational rfid. In *NSDI*, 2011.

[6] M. Buettner and D. Wetherall. An empirical study of uhf rfid performance. In *MobiCom*, pages 223–234, 2008.

[7] M. Buettner and D. Wetherall. A "gen 2" rfid monitor based on the usrp. *Computer Communication Review*, 40(3):41–47, 2010.

[8] M. Buettner and D. Wetherall. A software radio-based uhf rfid reader for phy/mac experimentation. In *RFID (RFID), 2011 IEEE International Conference on*, pages 134–141. IEEE, 2011.

[9] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman. Challenge: Ultra-low-power Energy-harvesting Active Networked Tags (EnHANTs). In *MobiCom*, 2009.

[10] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan. On the limits of effective hybrid micro-energy harvesting on mobile crfid sensors. In *MobiSys*, pages 195–208, 2010.

[11] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power Management in Energy Harvesting Sensor Networks. *ACM Transactions Embedded Computing Systems*, 6(4):32, 2007.

[12] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. E. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: a reliable bulk transport protocol for multihop wireless networks. In *SenSys*, pages 351–365, 2007.

[13] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Sensor Networks. In *Ambient Intelligence*. Springer Verlag, 2004.

[14] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched networks: A new paradigm for wireless transport. In *NSDI*, pages 423–436, 2009.

[15] B. Ransford, J. Sorber, and K. Fu. Mementos: system support for long-running computation on rfid-scale devices. In *ASPLOS*, pages 159–170, 2011.

[16] J. R. Smith, A. P. Sample, P. S. Powledge, S. Roy, and A. Mamishev. A Wirelessly-Powered Platform for Sensing and Computation. In *UbiComp*, 2006.

[17] J. Sorber, A. Kostadinov, M. Garber, M. Brennan, M. D. Corner, and E. D. Berger. Eon: A Language and Runtime System for Perpetual Systems. In *SenSys*, November 2007.

[18] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *TOSN*, 6(2), 2010.

[19] C. M. Vigorito, D. Ganesan, and A. G. Barto. Adaptive Control of Duty Cycling in Energy-harvesting Wireless Sensor Networks. In *SECON*, pages 21–30, 2007.

[20] R. Want. RFID Explained. In *Synthesis Lectures on Mobile and Pervasive Computing*. Morgan & Claypool Publishers, 2006.

[21] E. Welbourne, K. Koscher, E. Soroush, M. Balazinska, and G. Borriello. Longitudinal study of a building-scale rfid ecosystem. In *Proceedings of the MobiSys*, MobiSys '09, pages 69–82, New York, NY, USA, 2009. ACM.

[22] P. Zhang, J. Gummeson, , and D. Ganesan. Blink: A high throughput link layer for backscatter communication. In *MobiSys*, 2012.